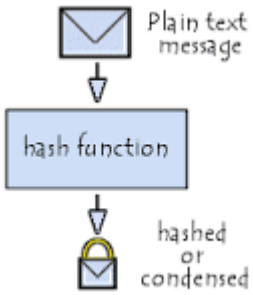
Table of Contents

Hash Function

1. A hash function is any function that can be used to map data of arbitrary size to data of fixed size. The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes.
2. One use is a data structure called a hash table, widely used in computer software for rapid data lookup. Hash functions accelerate table or database lookup by detecting duplicated records in a large file.
3. An example is finding similar stretches in DNA sequences. They are also useful in cryptography. A cryptographic hash function allows one to easily verify that some input data maps to a given hash value, but if the input data is unknown, it is deliberately difficult to reconstruct it (or equivalent alternatives) by knowing the stored hash value.
4. This is used for assuring integrity of transmitted data, and is the building block for HMACs, which provide message authentication.
5. Hash functions are related to (and often confused with) checksums, check digits, fingerprints, randomization functions, error-correcting codes, and ciphers. Although

these concepts overlap to some extent, each has its own uses and requirements and is designed and optimized differently.

6. Hash functions are primarily used in hash tables to quickly locate a data record (e.g., a dictionary definition) given its search key(the headword).
7. Specifically, the hash function is used to map the search key to an index; the index gives the place in the hash table where the corresponding record should be stored.
8. Hash tables, in turn, are used to implement associative arrays and dynamic sets.
9. Typically, the domain of a hash function (the set of possible keys) is larger than its range (the number of different table indices), and so it will map several different keys to the same index. Therefore, each slot of a hash table is associated with (implicitly or explicitly) a set of records, rather than a single record. For this reason, each slot of a hash table is often called a bucket and hash values are also called bucket indices.
10. Thus, the hash function only hints at the record's location — it tells where one should start looking for it. Still, in a half-full table, a good hash function will typically narrow the search down to only one or two entries.



Hash function has been defined using the following terms

Pre-image resistance

Given a hash value h it should be difficult to find any message m such that h = hash(m). This

concept is related to that of one way function. Functions that lack this property are vulnerable to preimage attacks.

Second pre-image resistance

Given an input m1 it should be difficult to find different input m2 such that hash(m1) = hash(m2). Functions that lack this property are vulnerable to second pre-image attacks.

Collision resistance

It should be difficult to find two different messages m1 and m2 such that hash(m1) = hash(m2). Such a pair is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for preimage-resistance; otherwise collisions may be found by a birthday attack.

Properties Of Hash Function

Determinism

A hash procedure must be deterministic, meaning that for a given input value it must always generate the same hash value. In other words, it must be a function of the data to be hashed, in the mathematical sense of the term. This requirement excludes hash functions that depend on external variable parameters, such as pseudo-random number generator or the time of day. It also excludes functions that depend on the memory address of the object being hashed in cases that the address may change during execution (as may happen on systems that use certain methods of garbage), although sometimes rehashing of the item is possible.

Uniformity

A good hash function should map the expected inputs as evenly as possible over its output range. That is, every hash value in the output range should be generated with roughly the same probability. The reason for this last requirement is that the cost of hashing-based methods goes up sharply as the number of collisions—pairs of inputs that are mapped to the same hash value—increases. If some hash values are more likely to occur than others, a larger fraction of the lookup operations will have to search through a larger set of colliding table entries.

Defined range

It is often desirable that the output of a hash function have fixed size (but see below). If, for example, the output is constrained to 32-bit integer values, the hash values can be used to index into an array. Such hashing is commonly used to accelerate data searches.

Variable range

In many applications, the range of hash values may be different for each run of the program, or may change along the same run (for instance, when a hash table needs to be expanded). In those situations, one needs a hash function which takes two parameters-the input data z, and the number n of allowed hash values.

Data normalization

In some applications, the input data may contain features that are irrelevant for comparison purposes. For example, when looking up a personal name, it may be desirable to ignore the distinction between upper and lower case letters. For such data, one must use a hash function that is compatible with the data equivalence criterion being used: that is, any two

inputs that are considered equivalent must yield the same hash value. This can be accomplished by normalizing the input before hashing it, as by upper-casing all letters.

Continuity

A hash function that is used to search for similar (as opposed to equivalent) data must be as continuous as possible; two inputs that differ by a little should be mapped to equal or nearly equal hash values.
Note that continuity is usually considered a fatal flaw for checksums, cryptographic hash function, and other related concepts. Continuity is desirable for hash functions only in some applications, such as hash tables used in Nearest neighbour search.

Related posts:

1. Types of Attack
2. Security threats
3. Computer and cyber security
4. Introduction to network security
5. Intrusion detection tool
6. Categories of security assessments
7. Security terminologies and principals
8. Intoduction to intrusion
9. Intrusion detection tool
10. Categories of security assessments
11. Intrusion terminology
12. Cryptography attacks
13. Cryptography
14. SSH

15. MD5
16. Message digest functions
17. Digital signature
18. Authentication Functions
19. One way hash function
20. Digital signature standard
21. SSL Secure socket layer